AUTOMATED RULE-BASE CREATION via CLIPS-INDUCE

Patrick M. Murphy

Department of Information & Computer Science
University of California, Irvine, CA 92717
pmurphy@ics.uci.edu
(714) 725-2111

Abstract

Many CLIPS rule-bases contain one or more rule groups that perform classification. In this paper we describe CLIPS-Induce, an automated system for the creation of a CLIPS classification rule-base from a set of test cases. CLIPS-Induce consists of two components, a decision tree induction component and a CLIPS production extraction component. ID3 [1], a popular decision tree induction algorithm, is used to induce a decision tree from the test cases. CLIPS production extraction is accomplished through a top-down traversal of the decision tree. Nodes of the tree are used to construct query rules, and branches of the tree are used to construct classification rules. The learned CLIPS productions may easily be incorporated into a large CLIPS system that perform tasks such as accessing a database or displaying information.

INTRODUCTION

Many CLIPS rule-bases contain one or more rule groups that perform classification. In this paper we describe CLIPS-Induce, an automated system for the creation of a CLIPS classification rule-base from a set of test cases. The rule-base created by CLIPS-Induce consists of two sets of rules, a set of user query rules to ask the user for any missing information necessary to perform classification, and a set of classification rules that is used to make the classification.

In the remainder of the paper, a detailed description of CLIPS-Induce and ID3 will be presented, followed by an analysis of CLIPS-Induce and list of potential extensions.

DESCRIPTION

In this section a description of CLIPS-Induce will be given, along with an example of its usage on a real-world problem, the Space Shuttle Landing Control problem. The goal of this classification problem is to determine whether the space shuttle should be landed manually or automatically.

CLIPS-Induce takes as input a set of test cases and returns two sets of CLIPS rules that perform user querying and classification. Each case is described in terms of a set of feature-value pairs. The same set of features is used for each case. An example case for the shuttle problem is given in Table I.

Table I: Example case from shuttle problem.

- Landing = manual
- Stability = stab
- Error = mm
- Sign = nn
- Wind = tail
- Magnitude = OutOfRange
- Visibility = yes

One feature is identified as the feature to be predicted given the values of the other features. For the shuttle problem, the feature *Landing* is to be predicted in terms of the features *Stability, Error, Sign, Wind, Magnitude and Visibility.*

A decision tree is constructed from the set of cases using the decision tree construction algorithm ID3. The tree constructed from the shuttle cases is shown in Figure 1. The decision tree is then used to construct the user querying and classification rule sets. The basic organization for CLIPS-Induce to presented in Figure 2.

Decision Tree Construction

A decision tree predicts the value of one feature in terms of the values of other features. The process by which a prediction is made using a decision trees is described below.

Using the decision tree in Figure 1, the value of the feature *Landing* will be predicted for the example case shown in Table I. Starting at the top (root) of the decision tree, the value of the feature *Visibility* is checked. Because the value is *Yes* the node at the end of the branch labeled *Yes* is next tested. Since the value for the feature *Stability* is *stab*, the *Error* node is next checked. Traversal of the tree continues down the *not(ss)* branch (because the value of Error is not *ss*), across the *mm* branch and finally down the *nn* branch to the leaf labeled *Manual.* The value for the *Landing* feature, predicted by the decision tree for this case is *Manual.*

ID3 is a decision tree construction algorithm that builds a decision tree consistent with a set of cases. A high-level description of the algorithm is shown in Table II. The tree

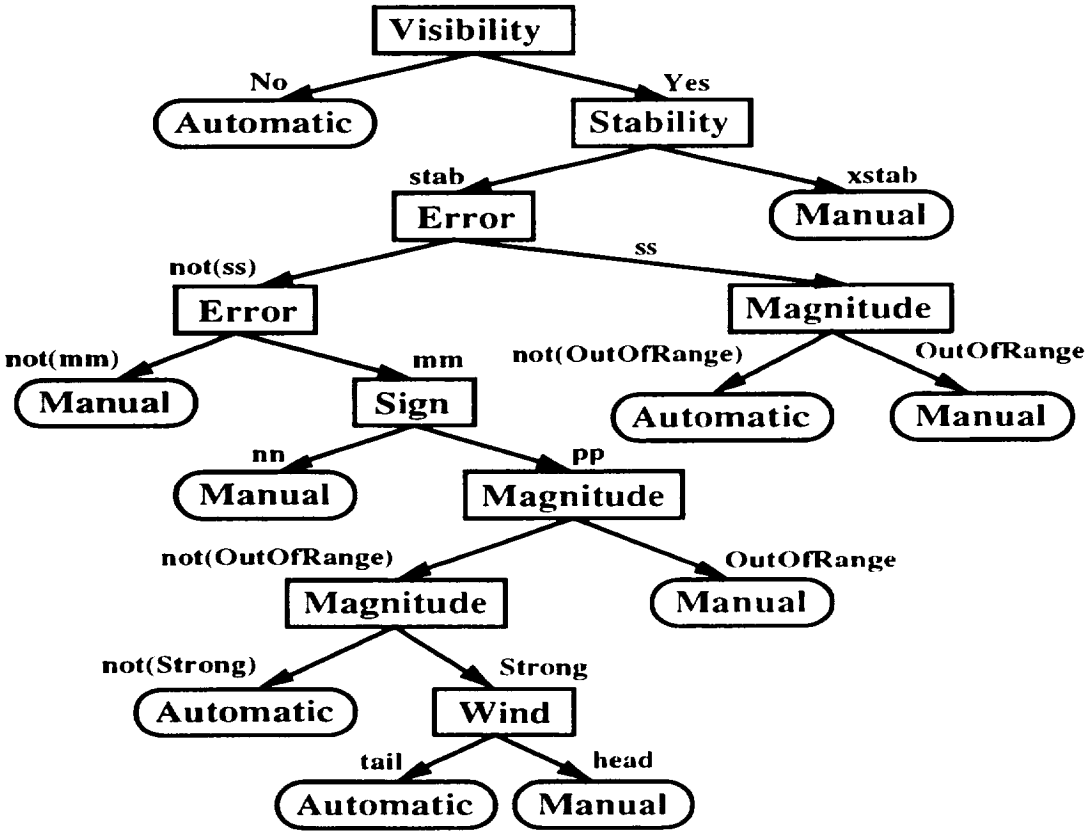

Figure 1: Decision tree constructed by ID3 using the shuttle cases.

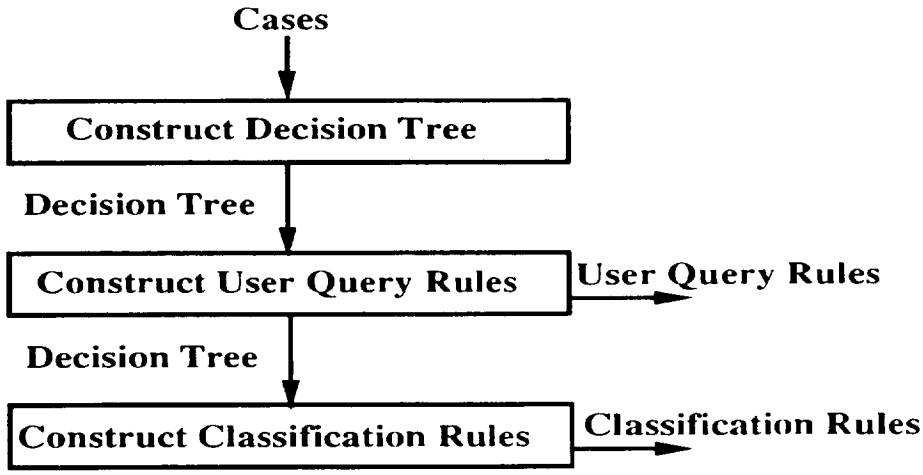

Figure 2: CLIPS-Induce Architecture

is constructed in a recursive top-down manner. At each step in the tree's construction, the algorithm works on the set of cases associated with a node in the partial tree. If the cases at the node all have the same value for the feature to be predicted, the node is made into a leaf. Otherwise, a set of tests is evaluated to determine which test best partitions the set of cases down each branch. The metric used to evaluate the partition made by a particular test is know as information gain (for a more complete description of information gain and ID3, see [1]). Once a test is selected for a node, the cases are partitioned down each branch, and the algorithm is recursively called on the cases at the end of each branch.

Table II: ID3 Decision Tree Construction Algorithm.

```
function generate_dtree(cases)
  if stop_splitting(cases)
    return leaf_class(cases);
  else
    best_cost := eval_examples_cost(cases);
    for all tests()
      cost := eval_cost(cases,test);
      if cost < best_cost then
        best_cost := cost;
        best_test := test;
    for all case_partitions(cases,best_test)
      branches := branches ∪{generate-dtree(case_partition)};
    return (best_test,branches);
```

There are three types of tests that are used by CLIPS-Induce to construct decision trees:

1. Two branch *feature* = *value* test: One branch for *feature* = *value* and a second branch for *feature* ≠ *value*.

2. Multi-branch *feature* = *value* test: A branch for each of the values that *feature* has.

3. Two branch *feature* > *value* test: One branch for *feature* > *value* and a second branch for *feature* ≤ *value*.

The first and second test types are used for nominal-valued features, e.g. color. Whereas the third test type is used for features with real or ordered values, e.g. age or size.

Rule Generation

The first step in rule generation is to generate the user query rules. The purpose of each user query rule is to ask the user for the value of a particular feature. The user-defined function used to ask the actual questions is shown below.

```
(deffunction ask-question (?question)
  (format t "%s " ?question)
  (read))
```

The query rules are generated such that they only fire when the value for a particular feature is needed and not already available. If, for example, the values for certain features were asserted before execution of the classification rules began, query rules for those features would never fire. Typically, user query rules and classification rules fire in an interleaved manner.

User query rules are generated via a pre-order traversal of the tree. During the traversal, each internal node of the tree is associated with a unique identifier that is used to identify the act of having visited that node during rule execution. An example user query rule, for the *Sign* node in Figure 1, is shown below.

```
(defrule sign-query-g773
  (node node8)
  (not (feature sign ?value))
  =>
  (bind ?answer (ask-question "What is the value of feature sign?"))
  (assert (feature sign ?answer)))
```

The second step in rule generation is to generate the classification rules. The purpose of the classification rules is to traverse the decision tree along a path from the root of the tree to a leaf. Upon reaching the leaf, the value for the feature to be predicted is asserted to the fact-list. Whereas query rules are associated with internal nodes in the decision tree, classification rules are associated with branches in the tree. The two rules for the branches from the *Sign* node in Figure 1, are shown in Table III.

The *Sign* node is identified as *node8*. If the value for feature *Sign* is *pp*, than *(node node9)* will be asserted by the first rule. *Node9* is associated with the *Magnitude* node. If the value for *Sign* were instead *nn*, the value *manual* for the predicted feature *Landing* would be asserted by the second rule. In the later case, because no new *(node ...)* fact is asserted, execution of the user query and classification rules halts.

Table III: Example classification rules.

```
(defrule node8-sign-pp
  ?n <- (node node8)
  (feature sign pp)
  =>
  (retract ?n)
  (assert (node node9)))

(defrule node8-sign-nn
  ?n <- (node node8)
  (feature sign nn)
  =>
  (retract ?n)
  (assert (feature landing manual)))
```

ANALYSIS

The first issue to be concerned with in using the CLIPS-Induce, is the time savings relative to generating the rules by hand. For the shuttle problem, the 25 rules were generated from a set of 277 cases in only a few seconds. For another problem that deals with predicting lymph node cancer, 87 rules were generated in less than a minute. Other problems have been observed to generate rule-bases with as many as 500 rules in very reasonable amounts of time. Given that a set of test cases is available, CLIPS-Induce can save a great deal of time.

The second issue concerns the accuracy of the induced rules on new cases. This concern has been addressed by the area of machine learning where a great deal of research has been done on the induction of decision trees from cases. Specifically, ID3 has been empirically shown to do well at generating decision trees that are accurate on unseen cases. For example, Figure 3 shows a learning curve for the shuttle problem. Learning curves show the accuracy of a model (a decision tree) on unseen cases, as a function of the number of cases used to generate the model. For the shuttle problem, when only 10% of the 277 cases were used to generate the decision trees, the accuracy on the remaining 90% of the cases is approximately 92%. As the proportion of cases increases, the accuracy of the constructed decision trees increases.

The third and final issue concerns the availability of a sufficient numbers of cases needed to induce an accurate set of rules (from Figure 3, the fewer the number of cases, the less accurate is the induced decision tree). In answer to this concern, even if there are only a small number of cases for a problem, the rule-base generated by CLIPS-Induce can be used as a starting point for a domain expert.

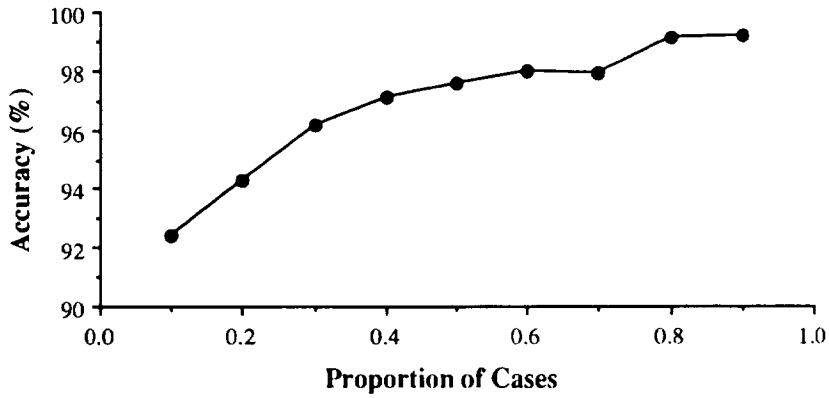

Figure 3: Average accuracy of decision trees as a function of the proportion of the 277 cases used to construct the decision trees. The accuracy of each decision tree is based on the cases not used to construct the tree.

EXTENSIONS

One of the ways that CLIPS-Induce could be extended would be to take advantage of ID3's approach for dealing with missing feature values. Currently, the rule-bases, generated by CLIPS-Induce, halt when the user cannot enter a value for a required feature. The only drawback to extending CLIPS-Induce in this manner, is the increased complexity and reduced understandability of the generated rules.

Another enhancement to CLIPS-Induce would be to use a more sophisticated *ask-question* function. User-query rules could be generated that also pass the set of allowable values or value type to the *ask-question* function. The extra argument could provide constraints on the allowable responses made by the user.

The third extension to CLIPS-Induce would be to allow interactive creation of decision trees. It is often the case that an expert in the field has knowledge that could help in forming a more accurate and more understandable decision tree.

CONCLUSION

In this paper, CLIPS-Induce, a Common Lisp application that induces a CLIPS classification rule-base from a set of test cases, is described. Given a set of test cases, described in terms of a fixed set of features, a decision tree is constructed using the decision tree construction algorithm, ID3. From the decision tree, two sets of rules are extracted. One set of rules, the user query rules, ask the user for the values of features needed to make a classification. The other set of rules, the classification rules, simulate a traversal of the decision tree in order to make the prediction that the decision tree would make. The rule-base formed by CLIPS-Induce can easily be embedded in rule-bases that need classification rule groups.

REFERENCES

1. Quinlan, J.R., "Induction of Decision Trees," MACHINE LEARNING, Boston, Massachusetts, 1(1), 1986, 81-106.